ERDC MSRC/PET TR/00-29

Enforcing Scalability of Parallel Comprehensive Mine Simulator (CMS)

by

Wojtek Furmanski
David Bernholdt
Geoffrey Fox

28 June 2000

Enforcing Scalability of Parallel Comprehensive Mine Simulator (CMS)

ERDC PET FMS Year 4 Focused Project Technical Report

Wojtek Furmanski, David Bernholdt, Geoffrey Fox *(contact person at fox@csit.fsu.edu)*
NPAC, Syracuse University

Syracuse, NY, June 2000

Introduction This project addressed the development of scalable Parallel CMS system by porting to Origin2000 the sequential CMS code developed by Ft. Belvoir. CMS is a substantial C++ code with a large number of dynamic objects and complex memory layout. Such codes are inherently hard to parallelize on current shared memory / NUMA platforms such as Origin2000 that are better tuned for conventional, more regular data parallel applications. In consequence, our initial attempts at Parallel CMS, based on standard Origin2000 semi-automatic parallelization techniques (such as compiler pragmas or OpenMP) were not very successful. Other FMS projects reported similar problems – for example the E-ModSAF effort within the CHSSI FMS-3 that aimed at porting ModSAF to Origin2000 was cancelled due to unmanageable complexity of dynamic memory layout of numerous simulation objects. To address the challenge of Parallel CMS we performed an in-depth analysis of the CMS source code, we experimented with a set of parallization techniques, and finally we succeeded in building a fully scalable Parallel CMS module. In this report, we summarize our approach and we present our performance, scalability and load balancing results.

Comprehensive Mine Simulator by Ft. Belvoir The Night Vision Lab at Ft. Belvoir, VA conducts R&D in the area of countermine engineering, using the advanced Comprehensive Mine Simulator (CMS) as an experimentation environment for a synthetic battlefield. Developed by the OSD sponsored Joint Countermine Advanced Concepts Technology Demonstration (JCM ACTD), CMS is state-of-the-art high fidelity minefield simulator with support for a broad range of mine categories, including conventional types such as buried pressure-fuzed mines, antitank mines and other types including offroute (side attack) and wide-area (top attack) mines. CMS organizes mines in components, given by regular arrays of mines of particular types. Minefields are represented as heterogeneous collections of such homogenous components. CMS interoperates via the DIS protocol with ModSAF vehicle simulators. Mine interaction with a target in controlled by its fuse. CMS supports several fuze types, including full width, track width fuzes, off-route fuzes and others. CMS mines can also interact with countermine systems, including both mechanical and explosive countermeasures and detectors.

The relevance of HPC for the CMS system stems from the fact that modern warfare can require a million or more of mines to be present on the battlefield, such as in the Korean Demilitarized Zone or the Gulf War. The simulation of such battlefield areas requires HPC support. As part of the PET FMS project, Syracuse University analyzed the CMS code and ported the system to the Origin2000 shared memory parallel MPP. Below, we summarize our approach and results.

Parallel CMS: Approach In our first attempt to port CMS to Origin2000, we identified performance critical parts of the inner loop, related to the repetitive tracking operation over all mines with respect to the vehicle positions and we tried to parallelize it using the Origin2000 compiler pragmas (i.e. loop partition and/or data decomposition directives). Unfortunately, this approach delivered only very limited scalability for up to 4 processors. We concluded that the pragmas based techniques, while efficient for regular Fortran programs, are not very practical for

parallelizing complex and dynamic object-oriented event driven FMS simulation codes - especially the 'legacy' object-oriented codes such as CMS which were developed by multiple programming teams over a long period of time and resulted in complex dynamic memory layouts of numerous objects that are now extremely difficult to decipher and properly distribute.

In the follow-on effort, we decided to explore an alternative approach based on a more direct, lower level parallelization technique. Based on our analysis of the SPEEDES simulation kernel that is known to deliver scalable object-oriented HPC FMS codes on Origin2000 (such as Parallel Navy Simulation System under development by Metron), we constructed a similar parallel support for CMS. The base concept of this 'micro SPEEDES kernel' approach, borrowed from the SPEEDES engine design but prototyped by us independently of the SPEEDES code, is to use only the fully portable UNIX constructs such as *fork* and *shmem* for the inter-process and inter-processor communication. This guarantees that the code is manifestly portable across all UNIX platforms, and hence it can be more easily developed, debugged and tested in the single-processor multi-threaded mode on sequential UNIX boxes.

In our micro-kernel, the parent process allocates a shared memory segment using *shmget()* and then it forks n children, remaps them via *execpv()*, and passes the shared memory segment descriptor to each child via the command line argument. Each child attaches to its dedicated slice of the shared memory using *shmat()*, thereby establishing the highest possible performance (no MPI overhead), fully portable (from O2 to O2K) multi-processor communication framework. We also developed a simple set of semaphores to synchronize node programs and to avoid race conditions in critical sections of the code. On a single processor UNIX platform, our kernel, when invoked with n processes, generates in fact n concurrent threads, communicating via UNIX shared memory. In an unscheduled Origin2000 run, the number of threads per processor and the number of processors used are undetermined (i.e. under control of the OS). However, when executed under control of a parallel scheduler such as MISER, each child process forked by our parent is assigned to a different processor, which allows us to regain control over the process placement and to realize a natural scalable implementation of parallel CMS.

Parallel CMS: Architecture On top of this micro-kernel infrastructure, we put suitable object-oriented wrappers that hide the explicit *shmem* based communication under the suitable higher level abstractions so that each node program behaves in fact as a sequential CMS, operating on a suitable subset of the full minefield. CMS module cooperates with ModSAF vehicle simulator running on another machine on the network. CMS continuously reads vehicle motion PDUs or the equivalent HLA interaction events from the network, updates vehicle positions and tracks all mines in the minefield in search for possible explosions. In our parallel version, the parent node 0 reads from the physical network and it broadcasts all PDUs via shared memory to children. Each child reads its PDUs from a virtual network which is a TCP/IP wrapper over the shmem communication channel.

Minefield segments are assigned to individual node programs using the scattered/cyclic decomposition which guarantees reasonable dynamic load balancing regardless of the current number and configuration of vehicles propagating through the minefield. We found the CMS minefield parser and the whole minefield I/O sector as difficult to decipher and modify to support scattered decomposition. We bypassed this problem by constructing our own Java based minefield parser using the new powerful public domain Java parser technology called ANTLR and offered by the MageLang Institute. Our parser reads the large sequential minefield file and chops it into n files, each representing a reduced node minefield generated via scattered decomposition. All these files are fetched concurrently by the node programs when the parallel CMS starts and the subsequent simulation decomposes naturally into node CMS programs,

operating on scattered sectors of the minefield and communicating via the shmem micro-kernel channel described above.

Parallel CMS: Performance We performed timing runs of Parallel CMS, using the Origin2000 systems at the Navy Research Laboratory in Washington, DC and at the ERDC Major Shared Resource Center at Vicksburg, MS. The performance results are presented in Figs. 1 and 2 and they illustrate that we have successfully constructed a fully scalable Parallel CMS for the Origin2000 platform. Figs. 1 and 2 present timing results of Parallel CMS for a large minefield of one million mines, simulated on 16, 32 and 64 nodes. The timing histogram in Fig. 1 displays total simulation times in a run on a 16-node spent by each of the nodes and it illustrates that we got almost perfect load balance. Higher bars on this figure represent full simulation run with all ModSAF PDUs activated, whereas lower bars represent dry CMS run without vehicle updates. The comparison of both sets illustrates that communication with ModSAF vehicles took of order of 20-25% of the total simulation time and that both computation and communication parts are fully load balanced.

Fig. 2 illustrates the speedup measured on 16, 32 and 64 nodes. Instead of T(1)/T(n) we present un-normalized 1/T(n) in this plot since we couldn't measure T(1) - when trying to run million mines simulation in one node we got memory overflow error. The SPEEDUP plot illustrates that Parallel CMS offers almost perfect (linear) scaling over broad range of processors.

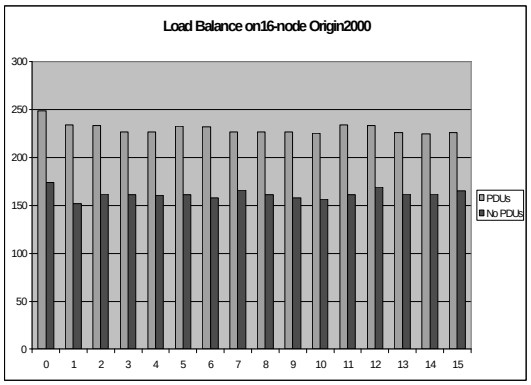

Fig. 1: *Simulation time spent by various nodes in a Parallel CMS run for million mines on a 16-node subset of Origin2000 at NRL (both for full run with vehicle PDUs and for a dry CMS-only run without PDUs) - illustrates very good load balance.*

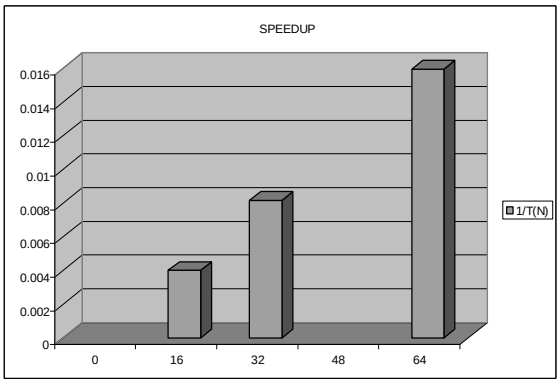

Fig. 2: *Speedup of Parallel CMS on NRL Origin2000 for million mines and 30 vehicles, measured on 16, 32 and 64 nodes - illustrates almost perfect scalability across a broad processor range.*

The timing results described above were obtained during Parallel CMS runs within our WebHLA [1][2] based HPDC / metacomputing environment that span four geographically distributed laboratories - ERDC in Vicksburg, MS, NRL in Washington, DC, ARL in Aberdeen, MD and NPAC in Syracuse, NY. We describe our WebHLA environment and the Metacomputing CMS runs in another Year 4 technical report [4] (see also the CRPC book chapter [3]).

Summary In this project, we demonstrated that the current generation of shared memory architectures such as Origin2000 can be successfully used not only for regular data parallel simulations but also for irregular, more dynamic and object-oriented modeling and simulation codes. Porting such codes cannot be accomplished by semi-automatic parallelization tools – it

requires more labor and more insight into the object memory layout of the sequential code. However, it appears that building scalable high performance codes for modeling and simulation is feasible and that the additional parallel code can be cleanly encapsulated from the legacy sequential code in the form of a suitable micro-kernel as described in this report. Scalable HPC M&S codes such as Parallel CMS and our micro-kernel based parallelization techniques described here, when combined with plug-and-play HLA based integration architecture (such as addressed in our other Year 4 project on WebHLA [4]) , can play an important role in facilitating HPC technology insertions into the new generation large scale M&S programs such as JSIMS, JMASS or JWARS.

References

1. Geoffrey C. Fox, Ph. D., Wojtek Furmanski, Ph. D., Ganesh Krishnamurthy, Hasan T. Ozdemir, Zeynep Odcikin-Ozdemir, Tom A. Pulikal, Krishnan Rangarajan, Ankur Sood, "Using WebHLA to Integrate HPC FMS Modules with Web/Commodity based Distributed Object Technologies of CORBA, Java, COM and XML," In Proceedings of the Advanced Simulation Technologies Conference ASTC 99, San Diego, April 99.

2. G. Fox, W. Furmanski, G. Krishnamurthy, H. Ozdemir, Z. Ozdemir, T.Pulikal, K. Rangarajan and A. Sood, "WebHLA as Integration Platform for FMS and other Metacomputing Application Domains," In Proceedings of the DoD HPC Users Group Conference, Monterey, CA, June 8-15, 1999.

3. CRPC Book Chapter, Morgan-Kaufmann (in progress): WebHLA based Metacomputing Environment for Forces Modeling and Simulation.

4. Wojtek Furmanski, David Bernholdt, Geoffrey Fox, "HLA Integration for HPC Applications Applied to CMS", ERDC MSRC PET Technical Report No. TR00-30, Vicksburg, MS, June 2000.